Data Storage Research Vision 2025

Report on NSF Visioning Workshop held May 30–June 1, 2018

George Amvrosiadis[†], Ali R. Butt[¶], Vasily Tarasov[‡], Erez Zadok[⋆], Ming Zhao[§]

Irfan Ahmad, Remzi H. Arpaci-Dusseau, Feng Chen, Yiran Chen, Yong Chen, Yue Cheng,
Vijay Chidambaram, Dilma Da Silva, Angela Demke-Brown, Peter Desnoyers, Jason Flinn, Xubin He,
Song Jiang, Geoff Kuenning, Min Li, Carlos Maltzahn, Ethan L. Miller, Kathryn Mohror, Raju Rangaswami,
Narasimha Reddy, David Rosenthal, Ali Saman Tosun, Nisha Talagala, Peter Varman, Sudharshan Vazhkudai
Avani Waldani, Xiaodong Zhang, Yiying Zhang, and Mai Zheng.

[†]Carnegie Mellon University, [¶]Virginia Tech, [‡]IBM Research,
[⋆]Stony Brook University, [§]Arizona State University

February 2019

Executive Summary

With the emergence of new computing paradigms (e.g., cloud and edge computing, big data, Internet of Things (IoT), deep learning, etc.) and new storage hardware (e.g., non-volatile memory (NVM), shingled-magnetic recording (SMR) disks, and kinetic drives, etc.), a number of open challenges and research issues need to be addressed to ensure sustained storage systems efficacy and performance. The wide variety of applications demand that the fundamental design of storage systems should be revisited to support application-specific and application-defined semantics. Existing standards and abstractions need to be reevaluated; new sustainable data representations need to be designed to support emerging applications. To take advantage of hardware advancements, new storage software designs are also necessary in order to maximize overall system efficiency and performance.

Therefore, there is a urgent need for a consolidated effort to identify and establish a vision for storage systems research and comprehensive techniques that provide practical solutions to the storage issues facing the information technology community. To address this need, the National Science Foundation's (NSF) "Visioning Workshop on Data Storage Research 2025" brought together a number of storage researchers from academia, industry, national laboratories, and federal agencies to develop a collective vision for future storage research, as well as to prioritize near-term and long-term storage research and scientific investigations. In-depth discussions were carried out at the workshop along four major themes: (1) Storage for Cloud, Edge, and IoT Systems; (2) AI and Storage; (3) Rethinking Storage Systems Design; and (4) Evolution of Storage Systems with Emerging Hardware. The participants especially underscored the need for focused educational and training activities to instill storage system tools and technologies in the next generation of researchers and IT practitioners. Finally, the development of shared, scalable, and flexible community infrastructure to enable and sustain innovative storage research and verifiable evaluation was also discussed. This report presents the findings from these discussions.

1 Introduction

There are a number of open challenges and research issues that need to be addressed both in the short and long term to ensure sustained storage systems efficacy and performance.

The wide variety of applications of modern and emerging storage systems demand that the fundamental design of storage systems should be revisited to support application-specific and application-defined semantics. Such tailored design will address many of the shortcomings of using the current one-size-fits-all generic approach that is plagued with inefficiencies both in performance and storage capacity overhead. Another aspect is to reevaluate the use of standards and APIs such as POSIX, and design new sustainable data representations and APIs to support emerging applications such as Internet of Things (IoT). Such redesign of the storage architecture is gradually being explored. For example, systems such as key-value (KV) and object stores are being used to design application-specific solutions. Management of metadata, indexing, and high-speed transactions for small data items are key to next generation storage systems as well. Nevertheless, newer methods and techniques are needed to support a wider range of emerging applications. This demands a more general discussion on application data management and storage system co-design.

Many important applications now arise in the area of Artificial Intelligence (AI), such as model training frameworks, where their needs are poorly met by current storage systems. New storage techniques and technologies tailored for AI workloads and AI usage of data are sorely needed. At the same time, AI-based methods hold great potential for building intelligent storage systems for meeting the challenging application demands and optimizing storage management as systems become increasingly complex.

Hardware advances are further driving the way storage systems are developed. Storage hybridization and heterogeneity is now a de facto part of most large-scale storage deployments. However, the software subsystems for supporting and using these advances are lagging, and new management systems need to be designed. Similarly, in-memory storage systems and persistent memory systems are giving rise to a new class of memory-only storage solutions. While such systems can be thought of as simply a tier of a traditional storage hierarchy, there is clearly a need for innovation to leverage the unique opportunities offered by new hardware advancements and designing the storage layer to maximize efficiency and performance. At the same time, the exponential growth of cold archival and cold primary data motivates intensive research into new hardware and software architectures that can store petabytes and exabytes of data at the best possible cost-efficiency trade-off.

More and more organizations are starting to trust computing clouds to cost-efficiently store and process data at scale. Public cloud providers now manage tremendous amounts of data; ongoing efforts to increase scalability of their infrastructures requires further innovation, both at the system software and hardware levels. Data management problems become especially challenging in hybrid clouds where the data flows need to be secure, tightly controlled, and frequently limited from being geographically or otherwise distributed. Furthermore, multi-tenancy at global scale increases workload and platform diversity to previously unseen levels—making the controlling of data-access quality an extremely complicated task with many open questions. At the same time, the dawn of the age of IoT is driving the need for novel and innovative storage systems at the edge of the Internet to store, manage, retrieve, and efficiently utilize unprecedented volumes of data at increasingly faster speeds.

Due to the aforementioned open challenges and issues, there is a need for a consolidated effort to identify and establish a vision for storage systems research and comprehensive techniques that provide practical solutions to the storage issues facing the information technology community. The goal of the National Science Foundation's (NSF) *Visioning Workshop on Data Storage Research 2025* was to bring together leading researchers in storage and distributed systems to provide a working vision, as well as prioritization for near-term and long-term storage research and scientific investigations. The workshop aimed to provide opportunities for identifying both core systems research in storage and cross-cutting research in "storage+X" systems.

The storage visioning workshop was held from May 31^{st} to June 1^{st} of 2018 at IBM's Almaden Research Center (ARC), a well-known industry research laboratory focusing, among other topics, on storage systems. The organizers invited about 50 researchers who are well-recognized leaders and key contributors to storage research across academia, industry, national laboratories, and federal agencies—to work together and develop a collective vision for future storage research. Every invitee was asked to provide a pre-workshop white paper on topics that they felt important. Based on a broad range of topics from the white papers, the organizers divided the workshop attendees into four groups to carry out focused discussions around these common major themes: **(i)** "Cloud, edge, and everything in between" (Section 2), **(ii)** "AI and Storage: Made for each other" (Section 3), **(iii)** "Teaching old storage new

tricks" (Section 4), and **(iv)** "The hardware, they are a-changin" (Section 5). In addition to research topics, workshop participants also discussed education issues, how to effectively teach the rapidly evolving storage technologies and grow the pipeline of researchers and developers who can contribute to future storage innovations (Section 6).

By the end of the workshop, each group also shared their discussion outcomes with all workshop participants to get feedback from the other groups and further develop the vision for each storage theme during the post-workshop, offline discussions. The rest of this report presents the findings on the aforementioned four major themes as the result of all of the pre-, during-, and post-workshop discussions.

2 Storage for Cloud, Edge, and IoT Systems

Data-driven scientific discovery has been well acknowledged as the new fourth paradigm of scientific innovation [53]. One key enabler of the data-driven innovation is data storage systems that manage the massive data, which witnessed a disruptive sea change in recent years. Among others, the advent of cloud computing has transformed the basic substrate for systems building in the last decade, and the long-anticipated "Internet of Things" (IoT) has led to the emergence of edge computing that extends the system boundaries pervasively.

In such a dynamic context, the depth of the storage stack and the scope of storage systems are increasing rapidly. Storage systems will need to manage data that is collected, stored, transformed, and transferred from heterogeneous edge devices to back-end cloud services, which can involve more than 18 layers [91]. There is a vast diversity of systems (e.g., different hardware, storage options, configurations); and at each layer of the system, different underlying storage technologies are likely to be the best fit for the corresponding workload and demands (e.g., latency, bandwidth, cost). Moreover, there are potential gaps or miscommunications between layers and components [47, 51], which increase the difficulty of providing end-to-end guarantees and achieving the ideal trade-offs among performance, reliability, fairness, etc. To move data storage research forward for cloud, edge, and IoT systems, we summarize the research challenges and opportunities into nine key properties that are essential for future storage systems, and call for the community's collective efforts to build systems that are truly efficient, unified, specified, elastic, explainable, sharable, application-driven, reliable, and re-evaluable.

In addition to research effort, the research infrastructure must also keep pace with the evolution of the computing landscape. We recognize that existing efforts such as CloudLab [3] help, but more are needed in order to sustain the advance of this area.

2.1 Research Challenges and Opportunities

2.1.1 Context: A World in Flux

The landscape of storage systems is changing rapidly. On the hardware side, there used to be just hard disk drives and tapes, but now we are seeing exciting advances including flash-based SSDs, byte-addressable persistent memories, and RDMA for storage networks (see Section 5 for more hardware-related discussions). In addition to superior raw performance, many of these new hardware systems are programmable to some extent and thus allow more customization.

Similarly, on the software side, there used to be just file systems. Now we have sophisticated key-value stores, document stores, configuration stores, logging stores, virtualized device mappers, etc.—many of which extend out to warehouse-scale cloud systems. (See Section 4 for more software-related discussions.) Moreover, virtualized cloud systems are able to manage the diverse resources (e.g., CPU cores, memories, SSDs) flexibly and transparently, usually scaling up and down within seconds [30].

One of the cloud's main benefits is that it hides much of the complexity from the users and offers greater convenience. However, the cloud's opaque, shared, diverse, and layered nature has raised concerns of how to provide end-to-end guarantees and achieve the ideal trade-offs among efficiency, elasticity, explainability, shareability, and reliability. We summarize the research challenges and opportunities into nine key properties that are essential for future storage systems. We elaborate on each of them next.

2.1.2 What Should We Build?

(1) Efficient systems. Similar to traditional systems, cloud-based systems also put great focus on efficiency. This is important for both users who pay for the usage (measured in hours/bytes/requests) and cloud service providers/builders who need to make profit. However, compared to traditional systems, there are many more layers involved in cloud systems: different layers usually require different data formats and read/write strategies to achieve the best local efficiency, which may conflict with other layers. Moreover, the diverse hardware, dynamic workloads, and the inherent multi-tenancy make achieving high efficiency even more difficult. Finally, more effort must be expended on capabilities that support Quality-of-Service (QoS) for end-to-end storage efficiency.

To address the challenge, we can no longer focus on a single layer or component. Instead, novel cross-layer and holistic end-to-end solutions are needed for removing all excess resource allocations in different layers, saving various costs (e.g., CPUs, memories, SSDs, energy), and achieving an overall high efficiency.

(2) Unified systems. Modern systems are filled with diverse storage options (e.g., various file systems, SQL databases, key-value stores, logging stores, configuration stores, document stores). While each individual storage option usually provides some unique features, they often have similar functions or components to some extent (e.g., managing persistent data structures). This inherent overlap of functionality is one of the major obstacles for building efficient systems today: for example, many physical copies of the same data exist across layers/components.

To address this challenge, we should explore the possibility of extracting the unified core components as building blocks and providing generalized solutions for various higher-level services. Also, to make different services more unifiable, we should experiment with solutions that can automatically transform configurations based on the dynamic needs of workloads: for example, addressing the underlying representation of data, the amount of resources allocated, and adapting configurations of durability and replication parameters.

(3) Specified systems. Current approaches to system building are too prescriptive, rigid, and error prone, which has led to various problems including downtime and data loss [51]; these inhibit scalability of future storage systems.

To address the challenge, we envision that future systems and applications should be specified in terms of performance requirements (e.g., which hardware feature is needed), persistence needs (e.g., which data needs to be stored where), and so on. Particularly, correctness properties should be precisely specified throughout the systems, which could potentially lead to the holy grail of verified systems that will never lose data. Along this direction, there are a number of open research questions though, including how to specify properties for the opaque cloud, how to identify the necessary properties and interfaces for each layer or component in the system, and how to specify the dynamic requirements of workloads.

(4) Elastic systems. Different from traditional storage clusters that are built on fixed hardware resources, cloud-based systems are naturally elastic. Such systems can be broken into constituent components that can be scaled up and down independently based on current workload demands.

We envision that the elasticity of the systems can be utilized for handling storage infrastructure tasks in addition to the workloads, which will likely improve the overall utilization and efficiency of the systems. To make better use of the elasticity for storage, more desegregated, composable software architectures are highly desirable. Instead of today's monolithic storage and file systems, we should experiment with different building blocks (e.g., unified core components) and micro-services, which can be reused across domains and improve long-term usability, reliability, etc.

(5) Explainable systems. Current cloud-based systems are opaque to the users. Many services are provided through relatively simple interfaces, which makes it difficult for users to reason about the provenance of their data. Moreover, due to the complicated layering within the cloud, it is also difficult for system builders or administrators to explain any observed abnormalities of system behaviors.

We envision that future systems should provide detailed provenance information—at a configurable provenance verbosity level—to allow for more explanations (e.g., how was a data object created, who can access what and why). This will be helpful for improving security (e.g., how information is leaked), reliability (e.g., how the data is corrupted), and performance (e.g., why this one run is slow).

(6) Sharable systems. Unlike the first-generation cloud technologies that only run a single or a few applications for one entity (person, organization), multi-tenancy is a new reality in modern cloud-based systems.

We believe one fundamental demand of multi-tenancy is effective sharing. However, achieving effective sharing is non-trivial as it involves many other aspects of systems. For example, from efficiency's perspective, multi-tenancy may cause interference among different workloads at different layers of systems, and thus violate QoS or Service Level Objectives (SLOs) for one or more users. Similarly, other security and privacy concerns (e.g., side-channel attacks, information leakage) need to be addressed more carefully in the multi-tenant environment in order to provide effective sharing.

(7) Application-driven systems. One major driving force of systems research is new application needs. There are many interesting new applications arising recently (e.g., videos from cameras at a soccer match, augmented reality applications), which place new demands on storage systems (e.g., real-time processing).

Given the vast diversity of applications, it is inefficient and impractical to build a highly specialized storage system for each application. Instead, we should avoid reinventing the wheel by exploring the commonality among applications and adapting storage systems for a range of applications automatically, similar to the design principle of unified systems. One unique challenge here is how to assemble a representative application suite and metrics for learning the common characteristics and demands.

(8) Reliable systems. As the scale and complexity of systems keep increasing, failures become the norm rather than exception [49]. Therefore, we need to design systems to deliver high performance and other desired properties in the presence of failures; this is particularly important for exabyte scale storage systems that will likely become common in 2025 [42].

Future systems need to be clearly specified, which could potentially lead to truly reliable storage that will never lose data. Existing efforts have shown that it is possible to formally specify and verify the crash consistency of one local file system built from scratch [41]. Nevertheless, how to scale the formal method to the vast majority of legacy software systems in the cloud environment remains unclear. More advanced mathematical methods and software engineering approaches (e.g., Netflix's Chaos Engineering [31]) are highly desirable.

(9) Re-evaluable systems. A constant theme in storage research is the availability of suitable workloads. This is critical for fair comparisons between systems and for generating reproducible results.

Unfortunately, compared to the workloads for local storage systems (e.g., Filebench, SPEC-SFS, and *many* more), far fewer workloads for cloud-based systems are publicly available. There are a number of workloads that have benefited the community (e.g., YCSB [43], Atlas [23]); but as systems keep evolving, much more representative workloads are needed to drive the research further. In a related issue, future storage systems should also be built with easy evaluation in mind (e.g., exporting internal performance metrics) to facilitate the fair comparison of design trade-offs under the same representative workloads.

2.1.3 Edge and its Impact on Cloud

Other than the cloud computing that has revolutionized the landscape of storage in the past decade, IoT is becoming a reality, bringing with it an explosion of data collection, storage, and processing demands. As predicted by Cisco, there will be over 50 billion IoT devices by 2020 [45]. The proliferation of IoT devices and the associated new demands have led to the emergence of edge computing. Essentially, the edge model places a "mini data-center" of compute and storage resources at the network edge, closer to the end users.

Compared with cloud computing, edge computing is much less mature or standardized. For example, there are heterogeneous devices manufactured by different vendors with various capabilities, protocols, and data formats. Also, the service models for IoT applications is unclear. We envision that one possible direction is the server-less computing model (e.g., AWS Lambda [1]). However, additional research efforts are needed to integrate the spectrum of diverse IoT devices into the current model.

Despite this heterogeneity, one common feature of all IoT devices is that they have limited hardware resources. To address this constraint, we should explore how to identify and discard unimportant data in a timely fashion, and how to balance among storage, preprocessing, and communication between IoT devices and cloud.

Cloud systems can be built for various workloads and adapt to demands on the fly. Conversely, edge computing has a large upfront cost to install edge nodes and a limited opportunity for ad hoc multiplexing at runtime. In other words, edge computing needs to be designed for the right workloads in the first place, and we need to identify these workloads and match them to storage capabilities precisely.

One barrier to storage research in the era of cloud-edge computing is that no edge-to-cloud holistic persistent data storage capabilities exist today. Therefore, a testbed involving both edge and cloud is highly desirable. Such a testbed will enable many research opportunities. For example, we can experiment with trade-offs of latency, cost, and persistence in light of edge computing applications. One reasonable assumption for this direction is that we will have increasingly larger resource budgets per unit costs as the devices' location gets closer to the cloud.

Another barrier is the lack of agreed-upon workloads and traces for evaluation and comparison of new research designs. A realistic workload trace needs to track requests to read and write data across all devices, edge nodes, and cloud servers, including operations that transform or aggregate the data. Recent work on distributed system tracing [20, 39] may provide the mechanism for collecting such traces, but the research community also needs to agree on a trace format (e.g., SNIA's DataSeries [24]), and strategies for replaying any such traces that are collected.

2.2 Infrastructure

As mentioned in Section 2.1.3, one challenge for research in this area is building a research testbed with a rich diversity of storage resources to enable experimentation with different trade-offs. Some existing efforts like CloudLab [3] and Chameleon [2] have helped facilitate cloud research; this is particularly valuable for small universities where local resources are fairly limited. Nevertheless, as cloud-edge systems keep evolving, the scale needed for realistic testing may be out of reach for many academic researchers, so larger-scale testbeds are highly desirable. Moreover, it would be helpful to provide standard services (e.g., S3, Lambda) along-side the testbeds to increase the reliability.

We envision a number of potential issues regarding shared testbeds. First, the typical chaos of shared testbeds is bad for science. It may be difficult to reproduce results in the shared cloud storage environment. Second, storage necessarily requires persistent data, and some workloads may lead to early wear-out of some storage technology. Both factors make it harder to share a storage research testbed, compared to similar shared infrastructure for networking or computing research. Therefore, techniques for building truly sharable and re-evaluable systems are urgently needed (see Section 2.1.2).

2.3 Near-term Industry Considerations

With the growing popularity of cloud computing among its customers, the industry is facing a number of immediate challenges. First, there are at least three options where to store and compute data: on-premises (also known as "private cloud"), in a public cloud, or using a hybrid approach; therefore, the choice of placing the workload is perplexing. Cloud service providers need efficient methods and tools to quickly evaluate where is best to place the workloads given the diverse and multi-dimensional customer requirements. Furthermore, the choice has to be future-proof and easily explainable to the customers. Second, as large enterprises and government institutions start to consider using public clouds, effective migration of massive amounts of data (into and out of clouds) is a striking challenge. Third, adopting new workloads, such as High-Performance Computing (HPC), in the cloud is out of the comfort zone for a typical cloud provider who uses commodity hardware for its infrastructure. The capacity to serve HPC workloads in the most cost-effective way is on the industry's list of near-term challenges. Fourth, as clouds adapt new technologies—both hardware and software-defined—the clouds' stability remains of utmost importance for the clients. The ability to rapidly integrate new technologies without disrupting the availability and reliability of the cloud is an important concern in the competitive cloud market.

3 AI and Storage

In this section, we discuss the intersection of Artificial Intelligence (AI) and Storage; we outline key research areas for storage in the context of AI applications, workloads, and trends. We explore this intersection as follows. We begin with a perspective on each (AI and Storage) independently, and then describe how each can benefit the other.

Storage for AI focuses on how storage research can drive systems designs to better serve AI workloads and AI usages of data. Conversely, AI for Storage focuses on how storage systems can be improved via internal application of AI techniques. Finally, we cover benchmarking challenges and industry considerations.

3.1 What is AI?

Artificial Intelligence (AI) is an umbrella term that covers a wide range of techniques to mimic human intelligence via machines and programs. AI includes the broad discipline of Machine Learning (ML) that itself includes a wide range of algorithmic techniques from statistical approaches, to neural network (Deep Learning) approaches, evolutionary algorithms, and more. At a high level, ML refers to the ability to model and extract patterns from data or observations (training) and subsequently use these models to make predictions on new observations (inference).

Although AI and ML have existed as separate fields for decades, the last 5–10 years have witnessed an exponential growth in the development and application of AI. Today, virtually all commercial industries are either applying or planning to apply AI techniques to enhance their respective disciplines. This shift is driven by three factors: data, compute, and algorithms.

(1) The Data. Devices such as sensors and robots (e.g., IoT) are generating increasing amounts of data and increasingly richer data, ranging from simple value time series to images, sound, and video. Although the data itself is valuable—its ultimate benefit to science or a business's bottom line comes from the analytics that extract the insights hidden within. Simple datasets such as streams of individual values can be analyzed via database queries or complex event processing techniques. However, the increasing richness of data (e.g., multiple correlated mixed type streams, images, sound, video) requires more complex ML and Deep Learning (DL) approaches. The increased volumes of data also enable ML/DL algorithms to achieve peak efficiency.

(2) The Compute. The ubiquity of high-performance commodity computing, driven by both massive core count increases in individual CPUs and low-cost cloud computing services, have made it possible to match data growth with similarly scalable ML and DL capabilities. Hardware innovations such as TPUs, GPUs, custom FPGAs, and instruction set support in modern CPUs have further improved ML algorithms' performance, making it practical to train using massive datasets.

(3) The Algorithms. The availability of open source algorithms for ML and DL has helped a lot, using libraries for analytic engines (e.g., Spark [95], TensorFlow [19], Caffe, NumPy [17], Scikit-learn [18]). With these packages, a wide range of algorithmic techniques are now available in the Data Scientist's sandbox. With open source, even the newest state-of-the-art algorithms in research are frequently and publicly available to test, tune and use, nearly as soon as they are invented.

The confluence of these three factors has fueled AI growth, and in turn will drive the need for combined storage and AI research.

3.2 Storage for AI

Storage technologies are likely to be more complex in the future, to support growing needs of big data and AI workloads, and really any workload that users need to store and process optimally. This complexity will demand support for different APIs at different levels. We expect to continue to see healthy use of block-level, file-level (e.g., POSIX), object, and key-value stores—and likely combinations thereof. There is a need for high-level, easy-to-use APIs that hide much of the internal complexity from users and developers; conversely, there is also a need to allow advanced (i.e., "power") users to access lower-level APIs, to enable more effective, custom optimizations. The key to the design of future storage systems and their APIs would be that they must be easy to use and logical for AI application developers and *at the same time* provide optimal storage at the lower levels (for any utility or cost function).

Specifically, the emerging AI field presents six trends that intersect with storage, where targeted storage research can benefit AI uses and AI applications.

(1) Massive datasets. AI workloads require the ingestion, preprocessing, and ultimately analysis of massive amounts of data. Multiple stages exist in typical AI pipelines, from data ingestion such as ETL (Extract, Transform, Load), to pre-processing (e.g., feature engineering, data wrangling, data cleansing/transformation), to the ultimate

execution of an AI algorithm in its training or inference phases. All of these can benefit from storage optimizations for performance and data management.

(2) AI stage awareness. Storage that is aware of the distinct stages or phases of AI processing can optimize AI pipelines via techniques such as caching of intermediate results, tracking of lineage, provenance, and checkpointing [46, 62, 72, 84].

(3) Compute architecture and data optimization. AI platforms follow distinct distributed computation architecture patterns (e.g., data parallel and model parallel [57, 95]). Memory hierarchy and data layout design for such computation patterns should also be a focus for future storage research. APIs that express the data access intent of an AI algorithm can also be a powerful tool to integrate memory hierarchy and data layout optimizations with AI computation.

(4) Unique characteristics. AI algorithms have unique characteristics that can be exploited to create efficient storage designs. Example characteristics include a tolerance to small amounts of data loss, very structured access patterns, and the ability to use and extrapolate from lossy compression.

(5) Access and distribution characteristics. Emerging access methods and characteristics associated with AI workloads, such as stream processing [9] or edge storage [12], also create unique challenges that should be focused on in future storage research.

(6) Security and compliance. The use of AI brings a new dimension to data security. As industries and users demand that decisions made by AI algorithms be reproducible, transparent, and explainable, pressure builds on enterprises to put in place data-management mechanisms to govern what data is and how it should be used to generate AI models and consequent insights [8, 10, 11, 35, 48].

3.3 AI for Storage

ML techniques should be researched to improve storage systems with respect to reliability, availability, and quality of service. The large and growing amount of available storage system historical access data will allow ML algorithms to be trained. Insights can be gained out of the training and thus used to help design or optimize storage systems in five ways.

(1) Performance and placement optimizations. ML algorithms can be applied to predict popular data and application patterns, which help improve various storage techniques, including tiered caching, prefetching, and resource provisioning. Adapting caching policy using online learning can have significant benefits: recent work [93] shows that using ML techniques to select between LRU and LFU replacement policies resulted in a significantly improved total cache hit rate under even smaller memory constraints. The key takeaway is that ML techniques are valuable for solving online optimization problems such as caching with the caveat that the primary knobs of control have to be orthogonal. We believe that ML can be applied with great success for other problems such as non-datapath server-side caches [36, 54, 58, 60, 70, 80, 83], distributed storage caches such as in hyper-converged systems, dynamic multi-tiered and hybrid storage systems [50, 64, 86, 94], and hybrid systems comprising DRAM and persistent memory.

(2) Failure prediction. Failure or error patterns in large storage systems, such as disk failures and silent data corruptions, can be predicted using ML techniques and correspondingly early detection and cautious measures can be taken to prevent errors from being propagated. For example, proactively replacing disks that are predicted to fail soon can reduce the cost of data loss or data rebuild.

(3) Storage tuning. Storage systems typically evolve to have a large number of tunable parameters. Parameters include hardware composition, I/O schedulers, tiering thresholds, cache sizes, and many more. Using learning and other black-box optimization techniques to advise administrators who build and maintain storage systems under dynamic workloads on the optimal parameter values could significantly improve system performance and cost for a given workload.

(4) Change and anomaly detection. Part of tuning for workloads is understanding when they change "phases," both temporarily and permanently. Anomaly detection has been an application area for ML techniques for over twenty years [59], and many techniques from these fields will likely translate to storage with little modifications.

The goals of these techniques are to recognize change over different time scales and assist with either debugging or re-tuning, as applicable.

(5) Intelligent storage devices within storage systems. Computing-enabled storage devices refer to storage devices with ample internal computing capability, typically also including some AI capabilities. On one hand, computing storage devices assume part of the storage-related computing functionality so that the running storage system is alleviated from excessive overheads. Therefore, the storage system can deliver improved performance. On the other hand, with more intelligent devices, researchers need to determine what level of intelligence is appropriate to offload to the device and propose techniques at the storage system level to achieve the best synergy.

The key challenge in the domain of using AI for Storage is that training data will often be limited before decisions have to be made. For instance, systems to store and quickly process data in self-driving cars must exist and run fast even *before* enough data can be collected for automated system design. Similarly, as storage needs shift over time in an organization, there may not be enough training data to predict how best to deal with changing priorities when reconfiguring factors such as parameters, tiers, placement, and layout. Tuning may also be improved by consideration of cost models outside of the standard throughput and latency optimizations of the bulk of storage research. Dollar cost, complexity, and power consumption come up as other potential reward functions in a multi-objective optimization scheme [65].

3.4 Benchmarks and Workloads

Since AI techniques are heavily data dependent, any strategy for driving AI and storage research needs to factor in the need for publicly accessible datasets and benchmarks. Public datasets exist in ML [25] but are in many cases too small to extract meaningful storage access patterns. Next, we describe three challenges that have to be overcome to drive expansive research into the storage and AI opportunities presented above.

(1) Dataset generation and collection. We need some systematic and sustainable schemes to generate and collect datasets, including: synthetic data generation of ML workloads; datasets from simulations; dataset gathered from prior NSF projects; and long-term data collection and dissemination via some community infrastructure such as NSF CRI/MRI.

(2) Characterizing workloads across layers. How to benchmark and characterize workloads from different layers including application, middleware, system, and storage-device layers is challenging and worthy of investigation.

(3) Workload classification. Classifying workloads has been studied for a long time [21, 71, 82]. As new storage platforms and applications are developed, there is a need to understand, in a way that is precise and communicable across different industries, what modern storage workloads look like. Given that a storage workload is a time series of operations, there are a variety of unsupervised as well as supervised ML techniques that we can apply to partition and categorize this space. We would use ML techniques to improve workload characterization in four areas: **(i)** Quantify similarity among workloads; **(ii)** Track changes in how a workload functions on a given architecture; **(iii)** Learn mixes of customer workloads on shared storage systems; and **(iv)** Detect phases of complex long-running workloads (e.g., scientific applications often run stages of data ingestion, data processing and output, and checkpointing).

3.5 Near-term Industry Considerations

The storage industry is attempting to capitalize on the popularity of AI and ML and provide specialized storage solutions for AI workloads [13, 16]. At the same time, storage companies are excited by the opportunities of using ML to improve performance and reliability, and develop quality products. In both cases, the main obstacle is the deficit of the professionals who are knowledgeable in both storage and AI areas. The number of fresh graduate students with this combination of skills is small, and training existing staff takes time and effort. Storage companies are also experiencing significant competition from other industries that require AI/ML knowledge (see also Section 6).

The near-term technical challenge for AI storage is the ability to provide enough bandwidth to powerful multi-GPU systems at an affordable price. Expensive systems like Lambda Hyperplane [14] and NVIDIA DGX-2 [15] are expected to be used for training ML models. Such systems can be equipped with 16 or more GPUs and process

hundreds of gigabytes of data per second. Appropriately designed storage is required to keep such expensive ML systems busy to ensure high efficiency and timely return on investment.

4 Rethinking Storage Systems Design

Six trends suggest that we need to fundamentally rethink the design of storage systems:

1. IoT and exascale High Performance Computing (HPC) clusters are both on track to produce a torrent of data far greater than the storage and network capacity of current systems.
2. The rapid growth of data science is introducing new workloads with unique storage access patterns and performance demands.
3. Current storage systems with minimal schemas are poorly equipped to organize huge amounts of data, especially when the growth in data size is projected to be exponential in nature.
4. Emerging storage technologies, such as DNA [34] and glass [74] storage, and storage-class memory, require rethinking the entire storage hierarchy from applications to hardware.
5. Privacy and security increasingly require methods to reason about the relationships among data being stored, as well as the provenance and lineage of the data.

In response to these trends, storage researchers will need to evolve current storage systems design. We identify five vital areas where research is needed:

1. Storage systems should allow far greater introspection into their operation and the data they store. We must develop methods to automatically use data from introspection to improve storage operation and improve data organization and management. Directions for future research in this area are covered in Section 4.1.
2. Storage systems should more tightly integrate computation (e.g., indexing, aggregation, transformation) with data generation and movement through the storage stack, effectively enabling "in-situ" and "in-transit" processing of data. The type of computation and the layer at which it is performed should adapt dynamically in response to changes in workloads and resources. Directions for future research in this area are covered in Section 4.2.
3. We should reconsider the fundamental design of the POSIX interface to support emerging storage technologies and use cases. This includes widening the interface, allowing more application-specific customization of storage behavior, and supporting evolution of data and technologies. Directions for future research in this area are covered in Section 4.3.
4. We should enable tighter co-design of applications and storage. This includes supporting "in-vivo" storage development, allowing evolution of storage systems and A/B testing, as well as rethinking where the "intelligence" of Flash storage should reside in the storage stack. Directions for future research in this area are covered in Section 4.4.
5. Finally, we discuss industry perspectives and considerations for this aspect of storage systems in Section 4.5.

4.1 Introspection, Provenance, and Metadata

4.1.1 Introspection

To maximize effectiveness, both storage systems researchers and storage systems themselves need to understand the detailed behavior of applications, storage devices, and the entire storage stack in between. To do that, we must be able to observe and correlate those behaviors both inside and outside the system.

There are currently many techniques for examining storage systems, some *ad hoc* and some semi-standardized [26, 29, 40, 73, 90]. However, future storage systems will demand improved approaches:

- The output of existing tools is generally designed to be consumed by humans rather than by programs, making the tools unsuitable for big-data analysis or for use in adaptive systems.

- Multi-layer storage systems are complex, making it difficult to correlate behavior at different levels. For example, a common problem occurs when two processes access the same file: Which should be "charged" for the resulting disk I/O? When an I/O request is merged with another, what is the underlying cause? Few, if any, tools address these issues effectively.
- Distributed and HPC systems have complex timing and subtle interactions [68]; capturing this distributed information is difficult.
- We do not know the best methods for analyzing traces. Traditional statistical tools seem unsuitable for describing complex behavior.
- We need to investigate new uses for introspection. Can modern file systems capture additional metadata to help users find data relevant to their needs? Can introspection at different levels of the storage stack be used for dynamic optimization?
- Real workloads often include multiple applications running simultaneously. Can we decouple the different signals to "tease out" the different applications [66]? Can we apply AI-like algorithms, such as those used to simulate the way the human ear can separate sounds in a noisy environment?
- Metadata about traces (e.g., characteristics of the traced system, details about workloads, environmental information) have traditionally been collected outside the system components that generated them and are usually stored independently of the traces themselves. This creates a disconnect that further complicates analysis.

We also need to better understand workloads and optimization in various ways, including:

- Can we build signatures for applications based on their I/O (and other) patterns?
- Can we generalize benchmarks to new storage technologies by using additional workload information? If so, what is the information necessary to achieve this?
- Can large-scale systems benefit from analyzing and modeling their workloads? Is it possible to understand and optimize data movement in these systems? For example, what semantics are needed for active processing with Flash?
- Can we provide applications with control of storage data layout by exposing user-defined functions [27]? Can we control data layout across the deep storage hierarchy as well as within it?

4.1.2 Provenance

An open research question is whether data provenance should be a first-class concern for storage systems. Current systems for tracking provenance are often external to the storage system; poor integration makes tracking provenance expensive and reduces the quality of provenance information. We note that provenance can be collected at multiple levels of fidelity: operations that produce data can be identified very generically (e.g., by application name), or the storage system can also include the arguments and parameters used to produce the data [75].

It is understood that to answer certain questions, a lot of provenance data may be necessary, which can easily exceed the size of the data about which the provenance is collected. Research is needed to investigate the trade-offs and right balance between how much provenance to collect and what questions can or cannot be answered with it. Put another way, it is challenging to determine the fidelity at which provenance information should be retained. If a query is not envisioned when the data is created, then provenance information required to answer that query might not be kept. The gold standard for provenance fidelity is reproducibility: if a storage system retains enough information to redo the original computation that produced the data, then it can answer arbitrary queries about how that data was produced by reproducing the original computation. Many original computations are deterministic; in this case, reproducibility requires retaining the original inputs [32, 52]. Otherwise, techniques that use deterministic record-and-replay can provide the needed reproducibility [44].

4.1.3 Metadata

The role of metadata is also poised to change in the future, larger-scale storage systems. As denser storage technologies become available, metadata will be of paramount importance in order to locate data that is relevant to users and

applications. As a result, we identify additional directions for future research, which complement the ideas proposed for introspecting storage systems and tracking data provenance.

At the time when data is generated, how it will be accessed, or what portions of it will be of most interest. In the case of provenance, we already discussed approaches that try to capture every facet of the data generation process. A similarly aggressive approach should be considered in metadata collection and generation. This calls for scalable approaches that treat metadata as a first-class citizen of the storage system. Several opportunities for future research arise if this paradigm is to be adopted. Maintaining large, metadata-rich namespaces that are both synchronous (i.e., immediately consistent in time) and provide high performance becomes challenging. Existing research examines approaches that reconsider and relax the requirement of a synchronous namespace [81, 97] in order to maximize performance.

Another challenge is determining what metadata to collect and how to collect it in a systematic way without affecting the performance of the foreground workload. Existing frameworks and tools that hook into layers of the storage stack may act as a starting point [22, 39]. In many domains, the data itself contains information that can be used to help answer data disposition questions, such as how to seamlessly extract metadata from data, how to create rich persistent indices, and how to derive more metadata. Traditionally, such metadata structures have been maintained outside of storage systems, causing inconsistencies to arise. Existing approaches [89] to metadata extraction and integration within file systems could serve as a starting point for approaching these questions. Finally, an implicit assumption when collecting metadata is that it will be accessible for an arbitrary amount of time. This can be challenging, and will require techniques that are capable of translating in-memory data structures into persistent, well-defined formats.

4.2 In-Situ and In-Transit Data Processing

The race to exascale clusters is ushering in an era where scientists generate massive amounts of scientific data for analysis, in the order of hundreds of terabytes per simulation [37, 38]. Meanwhile, private and government organizations amass user data into very large datasets used to train neural networks for any use case imaginable. This trend shows no signs of abating; it is partially driven by the rapid progress of research on denser storage technologies that can store this data [34, 74]. It is also caused by the steady growth in the number of ubiquitous sensors and IoT devices that generate data. Finally, there is a movement in systems modeling towards massive corpora of data, as opposed to domain expertise, likely due to the ease with which the former can be collected.

Today, scientific analysis is sped up through careful post-processing that optimizes the data layout on storage for expected access patterns. For other workloads, such as iterative machine learning and data analytics, the data can be distributed across multiple devices so that it can be accessed in parallel. However, neither of those approaches remains sustainable at scale, because as scientific output and datasets increase in size, the cost of post-processing or additional hardware management become prohibitive.

We encourage researchers to consider a paradigm where data is processed on its way to storage, i.e., while it transitions through the storage stack to become durable. The goal would be to drastically reduce (or perhaps even eliminate) the need for data post-processing. A number of directions for future research arise: *Where can computation be accommodated in the storage pipeline without impacting the primary workload? What are the types of computation that would be amenable or adaptable to be performed in the storage pipeline?*

4.2.1 Data Processing In-Situ and In-Transit

In-situ processing is a new paradigm for data processing that occurs during the application's runtime. Although promising, in-situ data processing can be challenging compared to post-processing. First, traditional post-processing programs are designed to use all the resources of the nodes they occupy. At the time an application runs, however, freeing up these resources to process data can negate the benefits of in-situ processing. Thus, in-situ processing must proceed with *limited resources*. Second, traditional post-processing programs assume full visibility of the data in order to partition it across devices. With in-situ processing, data must be handled in a streaming fashion, so *limited visibility* should be tolerated. Third, traditional post-processing programs are rarely expected to scale to thousands of nodes because storage or network I/O will usually be the bottleneck. With in-situ processing, however, such extreme scalability

may range from beneficial to necessary depending on the amount and type of resources unoccupied by the application.

One way to conduct in-situ analysis is to process the data on the storage system itself, where the data resides, e.g., on the storage servers or in the devices. Empirical studies have shown that there is slack on the storage servers or devices (e.g., SSDs) to conduct a limited amount of processing [67, 92]. Further, emerging flash devices have higher processing capability. Efforts such as Active Flash [92] and AnalyzeThis [88] are a starting point. Key research tasks include the construction of analysis abstractions within storage systems and devices, overlaying higher-level file system structures atop active flash devices, and placing data on the storage system in a manner that is conducive to future analyses.

Another way to process data on its way to storage is by running code in the network switches or other intermediate devices with limited computational capacity through which data flows, a technique we refer to as *in-transit* processing. Accelerators such as programmable NICs have been used recently to speed up in-memory key-value stores through caching [56, 61] and to provide strong consistency guarantees without compromising performance [63]. Continuing this trend to examine types of data processing that could be offloaded to such lightweight devices seems natural.

4.2.2 Types of Processing

A question that arises is whether there is computation that fits the constraints of the in-situ and in-transit processing models without special help, or whether current programs could be adapted to operate with such limited resources and visibility at extreme scale. Both could be potential directions for future research. Existing work has identified types of computation that is either amenable as-is, or can be adapted to the in-situ and in-transit models, such as debugging or monitoring tasks. Dapper [87] achieves scalable, unobtrusive tracing of distributed systems through filtering RPC requests so that the subset of messages retained form chains from source to destination. Performance monitoring can benefit from statistics collected in-situ or in-transit.

Recent work has shown that other tasks traditionally carried out by scanning the data could be adapted to the in-situ and in-transit models. The DeltaFS distributed file system [96, 97] constructs data indexes in-situ. This allows data queries to be drastically sped up with insignificant overhead at runtime, without the need for data post-processing. These results are also encouraging because they show that this can be achieved with frugal use of resources, leaving space for more processing. It thus becomes an interesting problem to find the limits of this approach by adapting other tasks, such as data analytics and modeling.

4.3 New Interfaces

There is a growing need to examine and redesign I/O interfaces for applications. The decades-old, legacy interfaces have proved functional but are reaching scalability limits in modern cloud, data center, and HPC areas. New interfaces have been tried before (e.g., within HPC [69]). But vendor lock-in fears have disincentivized adoption of new interfaces and instead have motivated very stable, narrow I/O interfaces with community-built middleware and file format-specific access libraries to work around specific shortcomings. However, due to these narrow interfaces, middleware and file formats have by design limited information about the performance characteristics of a particular storage system and therefore have to make a lot of assumptions.

With the "Cambrian explosion" of new storage devices with vastly differing performance characteristics, the above assumptions have become inadequate. Fortunately, the wide adoption of open-source software storage systems makes the adoption of new storage interfaces possible without raising the potential of vendor lock-in. Now is the time to evaluate existing interfaces to determine if modifying or extending them will enable scalability, as well as design new interfaces to explore new capabilities. For example, we may be able to relax POSIX semantics in well-defined ways such that we remove limitations, e.g., relax locking semantics in parallel systems. Alternatively, we may want to design new interfaces that give applications power to exploit the capabilities of upcoming storage hardware hierarchies, including storage class memory and NVRAM.

The motivation for re-examining and modifying legacy interfaces like POSIX is that they are widely used and a large number of existing applications depend on them. We need to understand what changes can be made to these legacy interfaces to address application scalability needs. However, this depends on having a clear understanding

of the I/O behavior of target workloads so that appropriate changes are made. For example, to determine if POSIX locking semantics can be relaxed in HPC workloads, we need to fully understand the access patterns of parallel applications in shared files. If no two processes write to the same offset in a shared file, then byte-range locking on the file may not be necessary. Other factors to consider are support of legacy interfaces like POSIX over modern storage infrastructures like object stores. We need to understand the inter-play between the storage models and explore methodologies for reducing overhead while continuing to support legacy applications.

New interfaces offer opportunities to present new capabilities to applications that have the potential to represent paradigm shifts in data management. For example, the current file-based paradigm encourages a static way of thinking about application data; when data changes, in many cases a new file is written to capture the change. However, if the fundamental concept in a storage system is no longer a file, but perhaps a data object that has a revision history as well as provenance information, the relationship of applications and users with data in storage becomes fundamentally different. As another example, legacy interfaces present a flat storage hierarchy to users, but modern and future storage systems are hierarchical and composed of fast, near-storage-like storage-class memory or NVRAM in addition to slower, permanent storage like a traditional parallel file system. New interfaces have the opportunity to expose aspects of a storage hierarchy for exploitation by applications. For example, if an interface allows an application to specify the persistence needs of a data object (perhaps as temporary), then the data object can be stored in the appropriate storage tier (e.g., temporary storage like NVRAM).

4.4 Co-Design of Applications and Storage

The traditional storage stack is organized as an hierarchical, multi-layer structure. As storage technologies become increasingly diversified (e.g., NAND flash, NVM, Optane), such a general-purpose layering structure falls short of being able to exploit the drastically distinct hardware properties.

A recent technical trend is to break the constraint of the storage layering structure by allowing software to obtain low-level control of hardware. For example, Open-Channel SSD [78] exposes device resources (e.g., I/O channels) to application software and allows applications to directly operate on physical flash media, enabling a variety of optimizations for applications [33, 55, 79, 85]. We envision such an integrative approach would become a more common practice, especially in light of emerging technologies. However, this practice fundamentally changes the way in which applications interact with storage, creating new research questions.

The first question is where such an application-device collaboration should reside. The device-level management (e.g., FTL) could be integrated with application's semantics at different levels. An aggressive method is to directly embed device-level management into the application logic. This approach removes the information barriers and maximizes the utilization of application's domain knowledge, but it causes a high dependency on hardware specifics. For certain application scenarios, such as HPC, customized in-house software, and bundled software/hardware products, this deep integration would be effective and affordable; for many others, however, it may not be suitable due to the higher development costs. A more conservative option is to place this management at the operating system level. For example, a specialized device driver can provide a system interface to receive semantic hints or advise from applications and then accordingly control the hardware. A third option is to provide a relatively shallow abstraction as a user-level library level. The library presents the storage as a set of APIs to allow application developers to obtain a fine-grained control over the device while still retaining a certain level of abstraction. Running at the user-level, the library development and debugging could also be simple and still portable.

Effective application-device co-design is non-trivial. It often tends to be ad hoc—each application is optimized individually. In fact, many optimizations needed across applications are often similar, if not exactly the same. It is desirable to identify a set of essential, sharable, low-level functions that are common across applications. For example, many applications want to allocate flash pages from a specified channel in an SSD. A set of such core functions could collectively form a new storage abstraction to facilitate co-design.

Cross-layer co-design complicates software development. Developers must consider both software logic and hardware issues. Debugging and testing will also be more complex, due to a larger test space. For example, a

software bug accidentally writing a Flash page twice without an erase may cause a silent data corruption. Methods to mitigate this added complexity are required.

4.5 Near-Term Industry Considerations

Although improving system performance with new application-specific interfaces is an effective approach, industrial solutions rely on standard interfaces to deliver compatible and well-tested systems. E.g., the support of the NFS protocol [76] will probably remain the major requirement for Network Attached Storage (NAS) appliances and services designed in 2025. Providing multiple interfaces—standard and application-specific ones—is a challenge as it increases the complexity of already complicated systems. The ability to effectively integrate multiple interfaces under the same hood is, therefore, a near-term challenge for the storage industry. A related research question is the design of the data schemas that are either optimal for multiple interfaces or can be efficiently transformed to suit different interfaces.

5 Evolution of Storage Systems with Emerging Hardware

Storage systems are poised for tremendous change due to rapid developments in both hardware technology and applications. In addition to continuous evolution of block storage in the form of enhanced SSD+HDD+SMR disks, storage architectures are impacted by a continuously shifting hardware ecosystem, triggered by advances in processor, memory, and networking technologies. Moreover, new applications like IoT and high-volume data analytics, coupled with shifting usage modalities based on data-center and cloud-based data management, are driving storage research to embrace a broad perspective incorporating architecture, operating systems, networking, compilers/languages, and applications.

5.1 Embracing Change: The New Normal

Storage systems occupy a central role within modern computer system architectures and their design is increasingly influenced by developments in the processor, memory, and network subsystems. Future storage systems will need to be flexible to respond to continual changes in the hardware ecosystem and agile in adapting to the evolving requirements of new applications. It is not an exaggeration to say that continual change in technology and applications is the new normal. Storage research should embrace methodologies and system designs that can take advantage of these advances in a modular, incremental, and timely manner. New storage technologies and applications spur novel storage system designs, that in turn can inform developments in future hardware components and software. We envision a dynamic and synergistic research agenda cutting across traditional systems boundaries and encompassing hardware architecture, operating system, compilers, distributed systems, and applications. The complementary challenges and opportunities for storage system research are elaborated in the two challenges described below.

5.1.1 Incorporate New Technology and Application Requirements

The changing landscape for storage systems is characterized by increasingly heterogeneous computing hardware, diverse, and less discontinuous memory hierarchies, novel interconnect and networking fabrics, and complex, highly dynamic workload patterns arising from applications like IoT and ML. The overarching challenge to deal with the constantly shifting landscape is to develop design methodologies that incorporate *dynamic policies and flexible interfaces that can adapt to new devices and workloads*.

For more than 60 years, the storage field was dominated by one type of technology, the magnetic spinning media or Hard Disk Drive (HDD). We believe that was an anomaly and that the future will be dominated by multiple, heterogeneous storage technologies and their combinations. There are many variants of Flash based devices, Storage-Class Memories (SCMs) and Non-Volatile Memories (NVMs), and even the HDD has been recast into several flavors of Shingled Magnetic Recording (SMRs) drives. Furthermore, there is active research on using glass [74] and DNA [34] as storage media. Hence, the time to investigate these ever-changing storage technologies is *right now*, when they are still relatively new and yet reasonably well understood.

We feel that future storage systems will most likely include a combination of two or more technologies. And, as NVMs begin to rival DRAM speeds, both will figure heavily in the deepening storage stack. Storage technologies, both volatile and non-volatile, may be arranged as in a tier or hierarchy, or any form of hybrid, cluster, or graph-like

configuration. Data may be cached or replicated in one or more tiers, or uniquely stored once in any single tier. Various policies will control when data is moved, copied, or cached between those tiers. We expect storage technologies to continue to evolve and change rapidly in the coming years, so an effective storage system has to be able to adapt to those changes quickly and efficiently (e.g., to optimize throughput, latency, energy, dollar cost, or any combination or complex utility function). In the following, we identify four trends that will play a key role in storage systems design.

(1) Memory and storage devices. The traditional world of fast, fine-grained volatile memory and slow block-level storage is in transition. New memory-bus-attached, byte-addressable non-volatile memory devices (B-NVM) are blurring the line between memory and storage. Potential applications of B-NVM range from adding another level in the storage hierarchy to eventual deployment as the central component of working storage. Meanwhile, block storage continues to evolve with new block NVM devices like Optane drives, devices with asymmetric read/write performance like SMR disks, and processing-in-storage devices like Kinetic Drives. Further diversification in terms of capacity versus performance-oriented devices and the tiering of data into active, working, near-line, cold, and archival levels, creates a complex storage landscape. Nascent technologies like DNA [34] storage raise intriguing possibilities for future archival storage at one end of the spectrum, while the large amounts of high-bandwidth memory close to the processor open up new opportunities for caching, buffering, and prefetching.

To exploit the advances in technology, research is needed to understand and control the evolving complex memory and storage hierarchy. Models, algorithms, and mechanisms need to be designed for data placement and data migration across multiple types of heterogeneous devices for the entire data lifetime. Issues of migration granularity and frequency, and analytical models must be addressed; the incorporation of ML techniques to automatically manage the hierarchy and meet QoS performance, reliability, availability, and consistency requirements must be developed.

(2) Network fabrics. Research is needed to understand the impact of new system interconnect technologies (e.g., CCIX, OpenCAPI), fast 40/100/200 Gbps Ethernet and 56/100/200 Gbps Infiniband networks, lightweight network fabrics (e.g., RDMA over Infiniband or RDMA over converged Ethernet or *RoCE*), and emerging NVM-motivated standards (e.g., GenZ) on storage system organization, and protocols for distributed and clustered storage. As the speeds of storage and networking converge, protocols designed for slow remote storage will need to be supplanted by more efficient schemes. An understanding of the design space and the networking abstractions required to support future storage systems is critical.

(3) Processor architecture. Heterogeneous processor architectures with specialized accelerators, GPUs, and FPGA devices can potentially facilitate compute-intensive tasks on behalf of the storage system. Examples include coding, compression, and support for cryptography and traffic management, to enable enhanced reliability, security, and QoS. Advanced processor mechanisms for concurrency control like Hardware Transaction Memory (HTM) can be leveraged to provide lightweight storage transactions in conjunction with fine-grained high-speed storage like B-NVM. Research to evaluate the potential of new processor hardware on storage and file system software, and the development of appropriate OS support is necessary to leverage processor architecture advances to benefit the next generation of storage systems.

(4) Applications. As data management becomes central to our everyday activities, storage systems will be increasingly called upon to support functionalities beyond the traditional role as performant suppliers of raw persistent data. The agility required of future storage systems to handle technology changes must carry over to designs that can incorporate new application-driven requirements gracefully and incrementally.

The imperative for security of stored data, verifiable access control, data privacy, and isolation will continue to increase. Data provenance and verifiable audit trails of data access and modification history over long data lifetimes will become increasingly necessary. Demand for quantifiable performance and QoS guarantees will increase for applications requiring time-sensitive response times and those deployed in shared data-center environments. Storage systems will need to satisfy micro-second and even nano-second latency requirements by carefully leveraging the memory hierarchy. Finally, application workloads will continue to change and increasingly include intricate, dynamic and fine-grained workload patterns for IoT, machine learning, graph analytics and other data-intensive high-performance computing tasks.

5.1.2 Storage-Driven Systems Research

Future storage systems will benefit from hardware and software support. Collaboration across various sub-disciplines is vital to providing a coherent path forward and avoid the reinvention of multiple overlapping solutions through silo-based research.

Two specific areas of synergy are in shaping the design of CPU architecture and OS abstractions, and the development of interconnect fabrics. Processor support is essential for using devices like B-NVM that have delays and reside between DRAM and typical storage. For instance, can processors better tolerate longer and more variable access delays (10–100s of microseconds)? What memory-mapping support is needed to handle fine-grained, local and remote terabyte-sized data? What hardware mechanisms to support transactions for persistent memory are appropriate? Should processors provide hardware logging and at what granularity? Is speculative paging worth supporting? In the networking domain, current RDMA protocols are inadequate to support end-to-end persistence and have scalability limitations. How should the storage system requirements drive the next generation of interconnection fabrics?

5.2 Impact of Byte-Addressable NVM

The availability of byte-addressable bus-attached NVM (B-NVM) devices gives rise to a new, potentially disruptive addition to the storage-memory hierarchy. B-NVM has advantages over DRAM in terms of capacity, power and non-volatility, and, in contrast to traditional block-structured storage, enables fast, cache-line-granular, direct processor access to persistent memory. The characteristics of B-NVM dovetail with modern applications handling huge persistent datasets and requiring fine-grained scattered data accesses, making it an excellent candidate for serving both memory and storage needs. Where B-NVM will ultimately fit in the storage hierarchy is currently an open question; its place will depend on the evolution of the cost, performance, and reliability characteristics of the underlying devices. There are intriguing possibilities and potentially large payoffs in terms of speed, power, crash resilience, and conceptual simplicity in deployment of B-NVM in the storage stack. To realize the potential of the new technology requires a strong research effort to deal with the multi-faceted and cross-cutting issues therein, which we discuss below.

5.2.1 Operating System and Application Development Support

The past decade has seen considerable research on the challenges of using B-NVM devices within a single server. The problems of consistent ordering between volatile and persistent memory, support for persistent memory transactions in the presence of failure, and coordination with volatile-memory concurrency controls mechanisms like HTM, have been addressed with solutions and techniques proposed in the architecture, OS, storage, languages, and database communities. Example important issues requiring research are identified below:

- **Security and integrity**: How to efficiently support encryption and low-latency access control at the granularity of memory accesses? How to avoid corruption of persistent data from malicious or buggy applications using lightweight mechanisms? Are language-level mechanisms appropriate or sufficient?
- **Abstractions for persistent data**: B-NVM can provide a single-level memory with a single namespace for both volatile and persistent data. What are the appropriate naming abstractions and required OS support? Are relative pointers the appropriate addressing mechanism? How should garbage collection be organized in this environment?
- **Consistency and transaction support**: Research is needed to transition the most efficient techniques into actual systems, and to build tools for building/migrating applications for direct memory access.

5.2.2 Distributed Shared NVM-Based Storage

While past research has concentrated on single-server-attached B-NVM, the next big step forward is understanding the challenges of deploying scalable, distributed B-NVM storage architectures in future data-centers. We discuss those in the following.

Distributed architecture. A basic issue is to understand the trade-offs in different architectural choices for distributed B-NVM. The design space includes distributed (persistent) memory systems, clustered storage systems, and hybrid architectures. The choices have significant impact on the usage modalities and on application structuring.

Careful analysis and empirical studies are necessary to understanding the trade-offs. This study should encompass the analysis of the underlying network fabric. Interconnect like Infiniband-RDMA or RoCE match the latencies of B-NVM devices making them a natural fit as the networking fabric. However, RDMA has known problems of scalability and has semantics that are unhelpful in providing the durability guarantees needed for B-NVM transactions. Understanding the correct abstractions that must be supported by the network fabric, and their impact on storage architectures and applications is needed to lay the foundation for future scalable B-NVM systems.

Software support. A strong research effort is needed in identifying the mechanisms and abstractions necessary to support future distributed, shared B-NVM. The problems are challenging and solutions are necessary to meet the scalability, reliability, usability, correctness, and latency requirements of applications. Many of these issues (listed below) have been previously examined in the context of single-server B-NVM systems, while distributed versions of these problems have been studied in the context of traditional block storage and TCP/IP networks. However, the distributed version of these problems deals with a far larger design space than server-attached B-NVM, and the micro-second-level latencies of B-NVM and direct-access RDMA-like protocols changes the nature of the problem qualitatively, requiring vigorous new research to explore the design space effectively. Some of the issues needing new research and lightweight solutions in the distributed B-NVM context are as follows:

- **Transparent global naming**: How should shared data be named, distributed, and accessed? What are the overheads for metadata management?
- **Transaction management**: How should transactions spread across multiple B-NVM hosts be orchestrated? How can RDMA-like protocols or atomics be exploited? What is the appropriate form of distributed logging?
- **Reliability and availability**: Whether replication or erasure coding be employed? How to support wear-leveling across nodes?
- **Consistency and coherence**: What are the appropriate consistency models for balancing application requirements and performance?
- **Crash resilience and durability**: How does one handle failures of nodes holding TBs of B-NVM data?
- **Metadata management**: How does one handle the size and overheads of metadata management?

5.3 New Storage Abstractions

New storage abstractions are necessary to support future storage architectures, new functionalities and modes of usage.

Processing in smart controllers. Research is needed to explore how to embed useful functionality in the path to storage through enhanced memory, storage, or network controllers. Current "bump-in-the-wire" functionality is provided by FPGAs at the back-end of NICs to provide on-the-fly transparent compression or encryption. Can offloading to dedicated controllers be useful in other contexts to provide support for functions like provenance, security, QoS, write ordering, or transaction support? For instance, an interposed persistent memory controller between the processor Last-Level-Cache (LLC) and B-NVM can simplify transactional programming. Can distributed persistent storage benefit from a controller at the NIC? Can functions like replication and data migration be offloaded to controller networks? Can stateful operations be successfully offloaded to controllers? Going beyond, what is the impact of embedding processing power within memory chips in processing-in-memory (PIM) models?

Programming models. As the abstraction levels of storage access rise, so does the need to define appropriate programming models and APIs for data access. What are the appropriate abstractions beyond traditional file system block reads and writes? Are KV stores supporting object access using GETs and PUTs adequate? Are transactional models useful in specifying application-level semantics? What are the boundaries between storage system, middleware, and applications? Will memory-style programming become the common mode for specifying persistent access? If so, what support will be needed from the compiler and programming languages? How would such specifications coexist and play with existing interfaces to block-based access?

5.4 Managing Exabyte-Scale Storage

Data reliability and durability. Proliferation and diversity of hardware causes an increase in the failure modes that need to be considered. The problem gets acute as the dataset scales requiring development of effective methods to handle ubiquitous failures in some portion of the storage system. The correlation between failure modes resulting in catastrophic data loss need to be addressed. The use of cross-layer reliability schemes to manage reliability overheads and exploit the availability of processing hardware provides a possible avenue of exploration. No one-size-fits-all storage solutions can exist for all different kinds of application workloads.

Data ingest and migration. As data-center storage becomes the de-facto technique to optimize data movement within data-centers, data migration across data-centers becomes a vital issue. Vendor lock-in inhibits the move to using a shared infrastructure despite other advantages; therefore, alternative approaches need to be considered.

5.5 Infrastructure Development

The rapid changes in hardware and application requirements present a tough challenge for researchers to evaluate their solutions. To allow for continued research and innovation, it is imperative to have access to platforms for experimentation and evaluation. One component would include the availability of shared hardware testbeds with state-of-the-art storage, networking and processing hardware. In addition, development of models and simulators for new devices, scalable system simulation and emulation tools, and collection and availability of modern workload traces are essential. For example, NSFCloud [4] is such a platform where two large-scale academic cloud testbeds—Chameleon [2] and CloudLab [3]—are providing researchers with a shared infrastructure to conduct scientific simulation and distributed system research validations at scale. However, as pointed out in Section 2.2, shared *storage testbeds* entail their own unique challenges that should be addressed. Additionally, a desirable feature of such open testbeds would be to open flexible options for more hardware configurations. For instance, emerging hardware devices such as Intel Optane SSD [5] can be integrated as an option available for storage and systems researchers, thus boosting cutting-edge storage research activities.

5.6 Near-term Industry Considerations

As new media such as Phase Change Memory (PCM) and SMR drives become available beyond prototypes and simulators (e.g., at industrial scale), the storage industry can start to evaluate their applicability in production. One of the main concerns for B-NVM is its early wear-out and unknown reliability properties, which may vary from one vendor to another. More efforts and research time is necessary to understand these unknowns.

Further, the ability to deploy and reuse existing enterprise, feature-rich storage software with next generation hardware is crucial. As a storage medium speed improves, the bottlenecks shift to CPUs and memory. Therefore, the software needs to be manually re-optimized for a new storage stack. This is a laborious and complex process. The issue becomes especially challenging when a lot of hardware combinations, each with different performance properties, are readily available.

6 Growing the Pipeline: Education and Teaching

In the past two decades, storage systems research has been increasingly hindered by a lack of practitioners. Demand has been increasing throughout computer science (CS), new areas of CS are drawing attention away from storage, and competition from industry causes large numbers of bachelor's graduates to discard research as a possible career. The problem is exacerbated by the fact that systems research in general, and storage in particular, is not highly visible to the public, is difficult to explain to the uninitiated, and suffers from a perception that the area is dull and overly technical. Everyone can understand and appreciate a robot that does a back flip; it is much harder to get the layman excited about a five-fold improvement in disk performance, let alone visualize it with exciting and enticing YouTube videos. Yet, the latter may have a much larger practical impact on our lives. In the following, we identify a number of aspects that can help the storage research community in sustaining and growing the pipeline.

(1) Mitigate diluted identity. Unlike other related communities who have well-established "home courses" (e.g., operating systems, computer networks, databases), storage systems are mostly taught under the hood of other courses.

Traditionally, storage systems are mostly taught as a sub-piece of operating systems. More recently, with the popularity of cloud and big data, many distributed storage systems are covered under broad topic courses such as cloud computing, and distributed and parallel database systems. Moreover, the novel storage hardware and devices are being incorporated into computer architectures, IoT, etc. Few universities teach storage in the name of "storage systems." Consequently, the identity of storage systems as a unique, standalone research area and community is diluted.

(2) Address underestimated scope. The program committee for USENIX FAST, the flagship storage systems conference, has a clear and broad definition [7]: "(we) will interpret 'storage systems' broadly; papers on low-level storage devices, distributed storage systems, and information management are all of interest." However, few students can interpret storage systems in this broad way.

Many students may only associate storage systems with hard disk drives or a specific file system, which is obviously less attractive compared to, say, self-driving cars. This situation is partly due to the fact that there is no clearly defined course on storage systems in the majority of universities.

To combat these perceptions and attract students into storage research, we need new approaches to education. We believe that presenting storage in a different light will increase the number of potential researchers and help address the crisis we are currently facing. One way to increase student interest is to give them direct, hands-on experience with storage systems. Multi-layer traces can be used for classroom demonstrations so that students can understand system operation and learn how to build large-scale storage systems.

(3) Provide cost-effective testbeds. A major challenge in teaching storage systems is that large systems (more than a few tens of terabytes and a few servers) are prohibitively expensive. A tool similar to networking's ns-2 [77] could help by providing students with a simulation platform for experimenting with large storage systems. Another possibility would be to create a community testbed similar to CloudLab [3], although such facilities are often still much smaller than real-world installations, and they tend to become overloaded at high-usage times such as early December when universities nationwide are finishing their academic terms. Such infrastructure becomes especially important if we are going to encourage a collaboration between data science and storage; we need compute platforms that are powerful enough to solve interesting problems on large-scale data. NSFCloud [4] is one of such early efforts to meet the demand.

Regardless of platform, students should have an opportunity to use and experiment with well-known storage systems and to build skills that are clearly useful in the field today, while still developing fundamental knowledge that will serve them well in the long term. For example, a system such as Hadoop could be an example of distributed file systems; students would be able to list that specific knowledge on their resumes but also could apply it to other areas.

(4) Provide course materials. We should increase the availability of ready-made courses, especially those from renowned universities, so that more researchers and educators would follow up. We should increase the availability of educational materials. Some existing efforts (e.g., OSTEP [28] and SNIA tutorials [6]) have helped a lot, but more efforts are needed to move the community forward.

(5) Improve storage research perception. One thing that attracts students to other areas is visibility: machine learning and deep learning are well covered in the press; gaming is something they're familiar with; robotics has long been a science-fiction dream; etc. We could improve storage's perception in the student community by increasing our use of visualization tools. Visualization can produce striking images that simultaneously attract the eye and provide deep insight. A class that asked students to visualize storage-related information could intrigue and challenge them. At the same time, it is difficult to visualize huge datasets that do not fit into memory; addressing that problem is a way to emphasize the importance of storage and engage students with the techniques needed to efficiently process out-of-memory data.

(6) Teach cross-disciplinary topics. To foster the collaboration between data science and storage systems, we need to equip students in storage systems with data science knowledge and vice versa. Nowadays, students are not well motivated to learn cross-disciplinary subjects. If high quality learning materials on data science are easily accessible from workshops, tutorials in systems conferences, and courses on massive open online courses (MOOCs), it would spark interest and encourage students to learn data science. Organizers of workshops should advocate research that

applies data science methodologies to solve problems of storage systems whenever applicable. Tutorials on data science related to systems are beneficial for students, researchers, and practitioners to quickly gain state-of-the-art methods and techniques of data science. Moreover, MOOCs-based courses nourish the community to embrace data science with low cost. There are also open source projects and contests held at conferences that offer opportunities for students to apply learned data science techniques to solve real world problems; these in return deepen the understanding of learned knowledge.

We should reach out to our colleagues in other sub-disciplines of computer science and partner with them. Data scientists can suggest new approaches to understanding storage systems, while we can help them manage vast quantities of information. Artificial intelligence and machine learning offer different approaches to optimization, and could be made more scalable and reliable. Library scientists understand how to organize information and might be able to suggest designs for presenting petabytes of data to users. Every community has a tendency to become insular; we should fight that tendency and actively seek collaborations that will be beneficial to both sides.

(7) Grow the number of storage researchers/academics. The above identified directions cannot be sustained without a corresponding investment in well-trained researchers and academics. We need to increase the number of academics working in storage systems, perhaps driven by increasing funding support from NSF and other agencies (e.g., similarly to the NSF's past HECURA program).

7 Conclusion

This report presents the discussions and perspectives of the storage community members that participated in the NSF-sponsored *Data Storage Research Vision 2025 Workshop* in May of 2018. We have identified four key thrusts: enhancing cloud and edge computing I/O infrastructures; designing storage for emerging AI applications; rethinking the storage systems abstractions in service of for new and innovative applications; and redesigning storage systems for emerging hardware. Focusing on these will help the storage research community address and mitigate the many identified and discussed challenges. Although there are other challenges not discussed here, our goal is to make the community aware of these and enable innovative research that can benefit the larger systems community, and in turn improve systems that underlie our modern life.

About this Document

This report summarizes the findings in the NSF "Workshop on Data Storage Research 2025," held in San Jose, CA on May 31–June 1, 2018. See **https://sites.google.com/vt.edu/data-storage-research/**. This was a two-day community visioning workshop to identify research challenges in designing novel and innovative storage systems to store, manage, retrieve, and efficiently utilize unprecedented volumes of data at increasingly faster speeds. It was organized by Ali R. Butt, Vasily Tarasov, and Ming Zhao. The workshop was open only to researchers who were invited and had submitted position papers in advance. About 40 researchers participated in the discussions and helped write this report. Participants came from academia, industry, and government, representing multiple storage, I/O, and distributed systems research communities. The workshop included invited presentations and in-depth discussions. The workshop program is listed as follows.

Thursday, May 31, 2018 (morning session):

08:00 Registration and breakfast (Building Lobby and Auditorium Foyer)
08:30 Introduction by workshop organizers (Auditorium)
08:45 Welcome message by NSF CSR Lead Program Director Dr. Samee Khan (Auditorium)
08:55 Overview of the NSF CSR Program by CSR Program Director Dr. Sandip Kundu (Auditorium)
09:30 Keynote talk: "Lessons learned storing 4 PB of telemetry data and transforming it into insights," Shankar Pasupathy, Technical Director (Analytics), NetApp Inc., (Auditorium)
10:30 Break (Auditorium Foyer)
10:45 Keynote talk: "Conjectures towards fruitful directions in data storage 2025," Irfan Ahmad, Co-founder and CEO, CachePhysics (Auditorium)
11:45 Keynote talk: "Building a future proof enterprise storage in 2025," Lawrence Chiu, Head of Storage Research and Steven Hetzler, IBM Fellow, Cloud Data Architecture, IBM (Auditorium)

Thursday, May 31, 2018 (afternoon session):

12:45 Lunch (Served out of J2-609 with seating in Cafeteria)
13:40 Introduction of group discussion topics (Auditorium):
AI and Storage: Made for each other (Room B2-425)
Cloud, edge, and everything in between (Room H2-214)
The hardware, they are a-changin' (Room G2-210)
Teaching old storage new tricks (Room J2-601)
14:00 Breakout group discussions on the topics led by the group moderators
17:30 Reception (Auditorium Foyer/Cafeteria Patio)

Friday, June 1, 2018:

08:00 Breakfast (Auditorium Foyer)
08:30 Summary presentations and report by each group (Auditorium)
10:00 Break (Auditorium Foyer)
10:30 Breakout into groups to discuss further and start the writing of group reports
11:30 Feedback and Open mic (Auditorium)
12:00 Adjourn (Boxed lunches provided) (Auditorium Foyer)

Acknowledgment

This material is based upon work supported by NSF under Award Number CNS-1829096). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] AWS Lambda: Serverless Computing. https://aws.amazon.com/lambda/. Accessed: June 6, 2018.

[2] Chameleon: A Configurable Experimental Environment for Large-Scale Cloud Research . https://www.cloudlab.us/. Accessed: June 6, 2018.

[3] CloudLab: Flexible, Scientific Infrastructure for Research on the Future of Cloud Computing. https://www.cloudlab.us/. Accessed: June 6, 2018.

[4] Enabling a new future for cloud computing. https://www.nsf.gov/news/news_summ.jsp?cntn_id=132377.

[5] Intel Optane SSD Series. https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/gaming-enthusiast-ssds/optane-900p-series.html.

[6] SNIA Education. https://www.snia.org/education. Accessed: June 6, 2018.

[7] USENIX FAST'18 Call for Papers. https://www.usenix.org/conference/fast18/call-for-papers. Accessed: June 6, 2018.

[8] SR 11-7: Guidance on Model Risk Management. Internet draft, 2011. https://www.federalreserve.gov/supervisionreg/srletters/sr1107.htm.

[9] Apache Flink. Internet draft, 2017.

[10] GDPR Online, 2017. http://www.eudgpr.org.

[11] New York Bill on Algorithmic Bias, 2017. http://www.businessinsider.com/algorithmic-bias-accountability-bill-passes-in-new-york-city-2017-12.

[12] The OpenFog Consortium: OpenFog Reference Architecture. Internet draft, 2017.

[13] DDN. https://www.ddn.com/, 2018.

[14] Lambda Hyperplane. https://lambdalabs.com/products/hyperplane, 2018.

[15] NVIDIA Dgx Systems. https://www.nvidia.com/en-us/data-center/dgx-systems/, 2018.

[16] Purestorage. https://purestorage.com, 2018.

[17] PyTorch, 2018. http://pytorch.org.

[18] scikit-learn, 2018. http://scikit-learn.org/stable.

[19] ABADI, M., AND ET AL. TensorFlow: Large-scale machine learning on heterogeneous systems. Internet draft, 2015.

[20] AGELASTOS, A., ALLAN, B., BRANDT, J., CASSELLA, P., ENOS, J., FULLOP, J., GENTILE, A., MONK, S., NAKSINEHABOON, N., OGDEN, J., RAJAN, M., SHOWERMAN, M., STEVENSON, J., TAERAT, N., AND TUCKER, T. The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Nov 2014), pp. 154–165.

[21] AGRAWALA, A. K., MOHR, J. M., AND BRYANT, R. M. An approach to the workload characterization problem. *Computer 9*, 6 (1976), 18–32.

[22] AMVROSIADIS, G., BROWN, A. D., AND GOEL, A. Opportunistic Storage Maintenance. In *Proceedings of the 25th Symposium on Operating Systems Principles (SOSP* (Monterey, California, 2015), ACM, pp. 457–473.

[23] AMVROSIADIS, G., PARK, J. W., GANGER, G. R., GIBSON, G. A., BASEMAN, E., AND DEBARDELEBEN, N. On the diversity of cluster workloads and its impact on research results. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)* (Boston, MA, 2018), USENIX Association, pp. 533–546.

[24] ANDERSON, E., ARLITT, M., MORREY, C., AND VEITCH, A. DataSeries: an efficient, flexible, data format for structured serial data. *ACM SIGOPS Operating Systems Review 43*, 1 (January 2009).

[25] ANGUITA, D., GHIO, A., ONETO, L., PARRA, X., AND REYES-ORTIZ., J. L. A Public Domain Dataset for Human Activity Recognition Using Smartphones. *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN* (2013), 1–15.

[26] ARANYA, A., WRIGHT, C. P., AND ZADOK, E. Tracefs: A File System to Trace Them All. In *Proceedings of the 3rd Usenix Conference on File and Storage Technologies (FAST)* (San Francisco, CA, Mar. 2004), Usenix Association, pp. 129–143.

[27] ARPACI-DUSSEAU, A. C., AND ARPACI-DUSSEAU, R. H. Information and Control in Gray-Box Systems. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)* (Banff, AL, Canada, Oct. 2001), pp. 43–56.

[28] ARPACI-DUSSEAU, R. H., AND ARPACI-DUSSEAU, A. C. *Operating Systems: Three Easy Pieces*, 0.91 ed. Arpaci-Dusseau Books, May 2015.

[29] AXBOE, J., AND BRUNELLE, A. D. blktrace User Guide. `http://www.fis.unipr.it/doc/blktrace-1.0.1/blktrace.pdf`, May 2008.

[30] BARROSO, L. A., CLIDARAS, J., AND HÖLZLE, U. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture 8*, 3 (2013), 1–154.

[31] BASIRI, A., BEHNAM, N., DE ROOIJ, R., HOCHSTEIN, L., KOSEWSKI, L., REYNOLDS, J., AND ROSENTHAL, C. Chaos Engineering. *IEEE Software 33*, 3 (May 2016), 35–41.

[32] BENT, J., THAIN, D., ARPACI-DUSSEAU, A. C., ARPACI-DUSSEAU, R. H., AND LIVNY, M. Explicit Control in a Batch-Aware Distributed File System. In *Proceedings of the 2004 Usenix Symposium on Network Systems Design and Implementation (NSDI)* (San Francisco, CA, March 2004).

[33] BJØRLING, M., GONZALEZ, J., AND BONNET, P. LightNVM: The Linux Open-Channel SSD Subsystem. In *Proceedings of the 15th Usenix Conference on File and Storage Technologies (FAST)* (Santa Clara, CA, 2017).

[34] BORNHOLT, J., LOPEZ, R., CARMEAN, D. M., CEZE, L., SEELIG, G., AND STRAUSS, K. A DNA-Based Archival Storage System. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (Atlanta, Georgia, USA, 2016), ACM, pp. 637–649.

[35] BURT, A. Is there a 'right to explanation' for machine learning in the GDPR. Internet draft, 2017.

[36] BYAN, S., LENTINI, J., MADAN, A., PABON, L., CONDICT, M., KIMMEL, J., KLEIMAN, S., SMALL, C., AND STORER, M. Mercury: Host-side flash caching for the data center. In *Proc. of IEEE 28th Symposium onMass Storage Systems and Technologies (MSST)* (April 2012).

[37] BYNA, S., SISNEROS, R., CHADALAVADA, K., AND KOZIOL, Q. Tuning Parallel I/O on Blue Waters for Writing 10 Trillion Particles. In *Cray User Group (CUG)* (2015).

[38] BYNA, S., USELTON, A., PRABHAT, D. K., AND HE, Y. Trillion particles, 120,000 cores, and 350 TBs: Lessons learned from a hero I/O run on Hopper. In *Cray User Group (CUG)* (2013).

[39] CARNS, P., HARMS, K., ALLCOCK, W., BACON, C., LANG, S., LATHAM, R., AND ROSS, R. Understanding and Improving Computational Science Storage Access Through Continuous Characterization. *Trans. Storage 7*, 3 (Oct. 2011), 8:1–8:26.

[40] CARNS, P., LATHAM, R., ROSS, R., ISKRA, K., LANG, S., AND RILEY, K. 24/7 Characterization of Petascale I/O Workloads. In *2009 IEEE International Conference on Cluster Computing and Workshops* (2009).

[41] CHEN, H., ZIEGLER, D., CHAJED, T., CHLIPALA, A., KAASHOEK, M. F., AND ZELDOVICH, N. Using Crash Hoare Logic for Certifying the FSCQ File System. In *Proceedings of the 25th Symposium on Operating Systems Principles* (New York, NY, USA, 2015), SOSP '15, ACM, pp. 18–37.

[42] CHIU, L., AND HETZLER, S. Building a future proof enterprise storage in 2025. *Data Storage Research 2025: An NSF-Sponsored Community Visioning Workshop* (2018).

[43] COOPER, B. F., SILBERSTEIN, A., TAM, E., RAMAKRISHNAN, R., AND SEARS, R. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing* (New York, NY, USA, 2010), SoCC '10, ACM, pp. 143–154.

[44] DEVECSERY, D., CHOW, M., DOU, X., FLINN, J., AND CHEN, P. M. Eidetic Systems. In *Proceedings of the 2014 Symposium on Operating Systems Design and Implementation* (Broomfield, CO, October 2014).

[45] EVANS, D. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. Cisco IBSG (Internet Business Solutions Group), 2011.

[46] FLINK, A. Checkpoints in Flink. Internet Draft, 2018.

[47] GANESAN, A., ALAGAPPAN, R., ARPACI-DUSSEAU, A. C., AND ARPACI-DUSSEAU, R. H. Redundancy Does Not Imply Fault Tolerance: Analysis of Distributed Storage Reactions to Single Errors and Corruptions. In *Proceedings of the 15th Usenix Conference on File and Storage Technologies* (Berkeley, CA, USA, 2017), FAST'17, USENIX Association, pp. 149–165.

[48] GARFINKEL, S., MATTHEWS, J., SHAPIRO, S., AND SMITH, J. Towards Algorithmic Transparency and Accountability. *Communications of the ACM, Vol. 60 No. 9, Page 5* (2016).

[49] GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. The Google File System. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2003), SOSP '03, ACM, pp. 29–43.

[50] GUERRA, J., PUCHA, H., GLIDER, J., BELLUOMINI, W., AND RANGASWAMI, R. Cost Effective Storage using Extent-based Dynamic Tiering. In *Proc. of the USENIX Conference on File and Storage Technologies* (February 2011).

[51] GUNAWI, H. S., HAO, M., SUMINTO, R. O., LAKSONO, A., SATRIA, A. D., ADITYATAMA, J., AND ELIAZAR, K. J. Why Does the Cloud Stop Computing?: Lessons from Hundreds of Service Outages. In *Proceedings of the Seventh ACM Symposium on Cloud Computing* (New York, NY, USA, 2016), SoCC '16, ACM, pp. 1–16.

[52] GUNDA, P. K., RAVINDRANATH, L., THEKKATH, C. A., YU, Y., AND ZHUANG, L. Nectar: Automatic Management of Data and Computation in Datacenters. In *Proceedings of the 2010 Symposium on Operating Systems Design and Implementation (OSDI)* (Vancouver, BC, Canada, October 2010).

[53] HEY, T., TANSLEY, S., TOLLE, K. M., ET AL. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research Redmond, WA, 2009.

[54] HOLLAND, D. A., ANGELINO, E., WALD, G., AND SELTZER, M. I. Flash Caching on the Storage Client. In *Proc. of USENIX ATC* (2013).

[55] HUANG, J., BADAM, A., CAULFIELD, L., NATH, S., SENGUPTA, S., SHARMA, B., AND QURESHI, M. K. FlashBlox: Achieving Both Performance Isolation and Uniform Lifetime for Virtualized SSDs. In *Proceedings of the 15th Usenix Conference on File and Storage Technologies (FAST)* (Santa Clara, CA, 2017).

[56] JIN, X., LI, X., ZHANG, H., SOULÉ, R., LEE, J., FOSTER, N., KIM, C., AND STOICA, I. NetCache: Balancing key-value stores with fast in-network caching. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)* (Shanghai, China, Oct. 2017), ACM, pp. 121–136.

[57] KIM, J. K., HO, Q., LEE, S., XUN ZHENG, W. D., GIBSON, G. A., AND XING., E. P. STRADS : A Distributed Framework for Scheduled Model Parallel Machine Learning.

[58] KOLLER, R., MARMOL, L., RANGASWAMI, R., SUNDARARAMAN, S., TALAGALA, N., AND ZHAO, M. Write Policies for Host-side Flash Caches. In *Proc. of USENIX FAST* (2013).

[59] LANE, T., AND BRODLEY, C. E. An application of machine learning to anomaly detection. In *Proceedings of the 20th National Information Systems Security Conference* (1997), vol. 377, Baltimore, USA, pp. 366–380.

[60] LEVENTHAL, A. Flash storage memory. *Commun. ACM 51*, 7 (July 2008), 47–51.

[61] LI, B., RUAN, Z., XIAO, W., LU, Y., XIONG, Y., PUTNAM, A., CHEN, E., AND ZHANG, L. KV-Direct: High-performance in-memory key-value store with programmable NIC. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)* (Shanghai, China, Oct. 2017), ACM, pp. 137–152.

[62] LI, H., GHODSI, A., ZAHARIA, M., SHENKER, S., AND STOICA, I. Tachyon: Reliable, Memory Speed Storage for Cluster Computing Frameworks. In *Proceedings of the ACM Symposium on Cloud Computing* (New York, NY, USA, 2014), SOCC '14, ACM, pp. 6:1–6:15.

[63] LI, J., MICHAEL, E., AND PORTS, D. R. K. Eris: Coordination-Free Consistent Transactions Using In-Network Concurrency Control. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)* (Shanghai, China, Oct. 2017), ACM, pp. 104–120.

[64] LI, Z., CHEN, M., MUKKER, A., AND ZADOK, E. On the Trade-Offs among Performance, Energy, and Endurance in a Versatile Hybrid Drive. *ACM Transactions on Storage (TOS) 11*, 3 (July 2015).

[65] LI, Z., MUKKER, A., AND ZADOK, E. On the Importance of Evaluating Storage Systems' $Costs. In *Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems* (2014), HotStorage'14.

[66] LIU, Y., GUNASEKARAN, R., MA, X., AND VAZHKUDAI, S. S. Automatic Identfication of Application I/O Signatures from Noisy Server-Side Traces. In *Proceedings of the 12th Usenix Conference on File and Storage Technologies (FAST)* (Santa Clara, CA, Feb. 2014).

[67] LIU, Y., GUNASEKARAN, R., MA, X., AND VAZHKUDAI, S. S. Server-side Log Data Analytics for I/O Workload Characterization and Coordination on Large Shared Storage Systems. In *Proceedings of the 29th International Conference on High Performance Computing, Networking, Storage and Analysis (SC)* (Salt Lake City, UT, Nov. 2016).

[68] MACE, J., ROELKE, R., AND FONSECA, R. Pivot Tracing: Dynamic Causal Monitoring for Distributed Systems. In *Proceedings of the 25th ACM Symposium on Operating System Principles (SOSP '15)* (Monterey, CA, Oct. 2015).

[69] MAY, J. *Parallel I/O for High Performance Computing*, 1st edition ed. Morgan Kaufman, 2000, ch. 4: "Access Patterns and Optimizations", pp. 87–89.

[70] MENG, F., ZHOU, L., MA, X., UTTAMCHANDANI, S., AND LIU, D. vCacheShare: Automated Server Flash Cache Space Management in a Virtualization Environment. In *Proc. of USENIX ATC* (2014).

[71] MESNIER, M., THERESKA, E., GANGER, G. R., ELLARD, D., AND SELTZER, M. File classification in self-* storage systems. In *null* (2004), IEEE, pp. 44–51.

[72] MIAO, H., LI, A., DAVIS, L. S., AND DESHPANDE, A. Towards Unified Data and Lifecycle Management for Deep Learning. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)* (April 2017), pp. 571–582.

[73] MICROSOFT CORPORATION. Event Tracing. `https://msdn.microsoft.com/en-us/library/windows/desktop/bb968803(v=vs.85).aspx`. Visited June 8, 2018.

[74] MICROSOFT RESEARCH. Project Silica. `https://www.microsoft.com/en-us/research/project/project-silica/`.

[75] MUNISWAMY-REDDY, K.-K., HOLLAND, D. A., BRAUN, U., AND SELTZER, M. Provenance-Aware Storage Systems. In *Proceedings of the Usenix Annual Technical Conference (ATC)* (Boston, MA, May/June 2006), pp. 43–56.

[76] NOWICKI, B. Nfs: Network file system protocol specification. Tech. rep., 1989.

[77] ns-2. `http://nsnam.sourceforge.net/wiki/index.php/Main_Page`. Visited June 8, 2018.

[78] Open-Channel Solid State Drives. `https://openchannelssd.readthedocs.io`, 2018. Visited June 8, 2018.

[79] OUYANG, J., LIN, S., JIANG, S., HOU, Z., WANG, Y., AND WANG, Y. SDF: Software-Defined Flash for Web-Scale Internet Storage Systems. In *Nineteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Salt Lake City, UT, Mar. 2014).

[80] QIN, D., BROWN, A. D., AND GOEL, A. Reliable Writeback for Client-side Flash Caches. In *Proc. of USENIX ATC* (2014).

[81] REN, K., ZHENG, Q., PATIL, S., AND GIBSON, G. IndexFS: Scaling File System Metadata Performance with Stateless Caching and Bulk Insertion. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* (New Orleans, Louisana, 2014), IEEE Press, pp. 237–248.

[82] RISKA, A., AND RIEDEL, E. Disk Drive Level Workload Characterization. In *USENIX Annual Technical Conference, General Track* (2006), vol. 2006, pp. 97–102.

[83] SANTANA, R., LYONS, S., KOLLER, R., RANGASWAMI, R., AND LIU, J. To ARC or Not to ARC. In *7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15)* (2015), USENIX Association.

[84] SCHELTER, S., BOESE, J.-H., KLEIN, J. K. T., AND SEUFERT, S. Automatically Tracking Metadata and Provenance of Machine Learning Experiments. *Workshop on ML Systems at NIPS (MLSys) Workshop.* (2017).

[85] SHEN, Z., CHEN, F., JIA, Y., AND SHAO, Z. DIDACache: A Deep Integration of Device and Application for Flash-based Key-value Caching. In *Proceedings of the 15th Usenix Conference on File and Storage Technologies (FAST)* (Santa Clara, CA, Feb. 2017).

[86] SHETTY, P. J., SPILLANE, R. P., MALPANI, R. R., ANDREWS, B., SEYSTER, J., AND ZADOK, E. Building Workload-Independent Storage with VT-Trees. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)* (San Jose, CA, February 2013), USENIX Association.

[87] SIGELMAN, B. H., BARROSO, L. A., BURROWS, M., STEPHENSON, P., PLAKAL, M., BEAVER, D., JASPAN, S., AND SHANBHAG, C. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Tech. rep., Google, Inc., 2010.

[88] SIM, H., KIM, Y., VAZHKUDAI, S. S., TIWARI, D., ANWAR, A., BUTT, A. R., AND RAMAKRISHNAN, L. AnalyzeThis: An Analysis Workflow-Aware Storage System. In *Proceedings of the 28th International Conference on High Performance Computing, Networking, Storage and Analysis (SC)* (Austin, TX, Nov. 2015).

[89] SIM, H., KIM, Y., VAZHKUDAI, S. S., VALLÉE, G. R., LIM, S.-H., AND BUTT, A. R. Tagit: An Integrated Indexing and Search Service for File Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* (Denver, Colorado, 2017), ACM, pp. 5:1–5:12.

[90] strace. `http://strace.io`. Visited June 8, 2018.

[91] THERESKA, E., BALLANI, H., O'SHEA, G., KARAGIANNIS, T., ROWSTRON, A., TALPEY, T., BLACK, R., AND ZHU, T. IOFlow: A Software-Defined Storage Architecture. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2013), SOSP '13, ACM, pp. 182–196.

[92] TIWARI, D., BOBOILA, S., VAZHKUDAI, S. S., KIM, Y., MA, X., DESNOYERS, P., AND SOLIHIN, Y. Active Flash: Towards Energy-Efficient, In-Situ Data Analytics on Extreme-Scale Machines. In *Proceedings of the 11th Usenix Conference on File and Storage Technologies (FAST)* (San Jose, CA, Feb. 2013).

[93] VIETRI, G., RODRIGUEZ, L. V., MARTINEZ, W. A., LYONS, S., LIU, J., RANGASWAMI, R., ZHAO, M., AND NARASIMHAN, G. Driving Cache Replacement with ML-based LeCaR. In *10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18)* (2018), USENIX Association.

[94] WU, X., AND REDDY, A. L. N. Exploiting Concurrency to Improve Latency and Throughput in a Hybrid Storage System. In *Proc. of IEEE MASCOTS* (September 2010).

[95] ZAHARIA, M., XIN, R. S., WENDELL, P., DAS, T., ARMBRUST, M., DAVE, A., MENG, X., ROSEN, J., VENKATARAMAN, S., FRANKLIN, M. J., GHODSI, A., GONZALEZ, J., SHENKER, S., AND STOICA, I. Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM 59*, 11 (Oct. 2016), 56–65.

[96] ZHENG, Q., AMVROSIADIS, G., KADEKODI, S., GIBSON, G. A., CRANOR, C. D., SETTLEMYER, B. W., GRIDER, G., AND GUO, F. Software-defined Storage for Fast Trajectory Queries Using a DeltaFS Indexed Massive Directory. In *Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)* (Denver, Colorado, 2017), ACM, pp. 7–12.

[97] ZHENG, Q., REN, K., GIBSON, G., SETTLEMYER, B. W., AND GRIDER, G. DeltaFS: Exascale File Systems Scale Better Without Dedicated Servers. In *Proceedings of the 10th Parallel Data Storage Workshop (PDSW)* (Austin, Texas, 2015), ACM, pp. 1–6.